**You said:**

some shell script related tasks that can be very important and used in real world please

**ChatGPT said:**

Here are some useful shell script tasks that can be important in the real world:

1. **Automating System Backups:**
   * You can create shell scripts to automate regular backups of important directories and files, ensuring data safety without manual intervention.

bash

CopyEdit

#!/bin/bash

# Backup script example

SOURCE\_DIR="/home/user/data"

BACKUP\_DIR="/backup/$(date +'%Y-%m-%d')"

mkdir -p $BACKUP\_DIR

cp -r $SOURCE\_DIR $BACKUP\_DIR

echo "Backup completed at $(date)" >> /var/log/backup.log

1. **File Management and Cleanup:**
   * Automate the removal of old or temporary files that are no longer needed, freeing up space and keeping the system tidy.

bash

CopyEdit

#!/bin/bash

# Remove files older than 30 days

find /home/user/temp/ -type f -mtime +30 -exec rm -f {} \;

1. **Monitoring Disk Usage:**
   * Create a script to monitor disk usage and send alerts if the disk space usage exceeds a certain threshold.

bash

CopyEdit

#!/bin/bash

# Monitor disk space usage

THRESHOLD=90

CURRENT\_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

if [ $CURRENT\_USAGE -ge $THRESHOLD ]; then

echo "Disk space is above threshold: ${CURRENT\_USAGE}%" | mail -s "Disk Space Alert" admin@example.com

fi

1. **Log Rotation:**
   * Set up automated log rotation to manage and compress logs, ensuring they don’t consume too much disk space.

bash

CopyEdit

#!/bin/bash

# Rotate and compress logs

LOG\_DIR="/var/log/myapp"

TIMESTAMP=$(date +'%Y%m%d')

find $LOG\_DIR -name "\*.log" -mtime +7 -exec gzip {} \;

mv $LOG\_DIR/\*.log.gz $LOG\_DIR/archive/$TIMESTAMP/

1. **Scheduled System Updates:**
   * Automate the process of updating the system and installed packages.

bash

CopyEdit

#!/bin/bash

# Automate system updates

sudo apt update -y && sudo apt upgrade -y

echo "System updated at $(date)" >> /var/log/system\_updates.log

1. **Automated Email Notifications:**
   * Send automated emails for system status reports, alerts, or important updates.

bash

CopyEdit

#!/bin/bash

# Send system status email

SYSTEM\_STATUS=$(uptime)

echo "System Status: $SYSTEM\_STATUS" | mail -s "System Health Report" user@example.com

1. **Service Restart on Failure:**
   * Automatically restart services like web servers if they stop working or crash.

bash

CopyEdit

#!/bin/bash

# Check if Apache service is running, if not restart

if ! systemctl is-active --quiet apache2; then

systemctl restart apache2

echo "Apache service was down and has been restarted" | mail -s "Service Restart Alert" admin@example.com

fi

1. **User Management:**
   * Automate the process of adding or removing users, and managing permissions.

bash

CopyEdit

#!/bin/bash

# Create a new user and set up directories

USERNAME="newuser"

useradd -m $USERNAME

mkdir /home/$USERNAME/logs

chmod 755 /home/$USERNAME/logs

1. **Scheduled Database Backups:**
   * Automate the backup of databases (e.g., MySQL) and save them with a timestamp for later retrieval.

bash

CopyEdit

#!/bin/bash

# Backup MySQL database

DB\_NAME="my\_database"

DB\_USER="root"

DB\_PASSWORD="password"

BACKUP\_DIR="/backup/mysql"

DATE=$(date +'%Y%m%d')

mysqldump -u $DB\_USER -p$DB\_PASSWORD $DB\_NAME > $BACKUP\_DIR/db\_backup\_$DATE.sql

1. **Automation of File Synchronization:**
   * Use rsync to automatically synchronize files between different machines or directories.

bash

CopyEdit

#!/bin/bash

# Sync files from local directory to remote server

rsync -avz /local/directory/ user@remote:/remote/directory/

These tasks are practical in the real world, whether you are managing servers, automating backups, or maintaining system health and performance.

4o

**You said:**

continue please

**ChatGPT said:**

Sure! Here are more real-world shell script tasks that can be highly useful:

1. **Automating System Cleanup (Temporary Files and Cache):**
   * Regularly clean up system caches, old logs, and temporary files to ensure the system stays lean and performs optimally.

bash

CopyEdit

#!/bin/bash

# Clean up system cache and temporary files

sudo apt-get clean

sudo rm -rf /tmp/\*

sudo rm -rf ~/.cache/\*

sudo journalctl --vacuum-time=7d # Remove logs older than 7 days

1. **Automated Website Backup:**
   * If you are managing websites, automate the backup of website files and databases.

bash

CopyEdit

#!/bin/bash

# Backup website files and MySQL database

WEBSITE\_DIR="/var/www/html/mywebsite"

DB\_NAME="website\_db"

DB\_USER="root"

DB\_PASSWORD="password"

BACKUP\_DIR="/backup/website"

DATE=$(date +'%Y%m%d')

# Backup files

tar -czf $BACKUP\_DIR/website\_files\_$DATE.tar.gz -C $WEBSITE\_DIR .

# Backup database

mysqldump -u $DB\_USER -p$DB\_PASSWORD $DB\_NAME > $BACKUP\_DIR/website\_db\_$DATE.sql

1. **Automating Server Reboots and Maintenance:**
   * Automate a regular reboot or restart of services on a server to ensure that the system runs smoothly.

bash

CopyEdit

#!/bin/bash

# Reboot the server and restart critical services

sudo reboot

1. **Automated SSL Certificate Renewal:**
   * Renew SSL certificates (e.g., from Let’s Encrypt) automatically and reload the web server to apply them.

bash

CopyEdit

#!/bin/bash

# Renew SSL certificates

sudo certbot renew

sudo systemctl reload apache2 # Or nginx

1. **Automating Application Deployment:**
   * If you're deploying applications, you can automate the process of pulling the latest code from a repository and restarting services.

bash

CopyEdit

#!/bin/bash

# Deploy the latest code from Git repository

cd /path/to/your/application

git pull origin main

sudo systemctl restart yourapp.service

1. **Automated System Health Monitoring and Reporting:**
   * Monitor the health of key system resources (e.g., CPU, memory, load) and send email notifications when critical thresholds are exceeded.

bash

CopyEdit

#!/bin/bash

# Monitor CPU, memory, and disk usage

CPU\_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.\*, \*\([0-9.]\*\)%\* id.\*/\1/" | awk '{print 100 - $1}')

MEM\_USAGE=$(free | grep Mem | awk '{print $3/$2 \* 100.0}')

DISK\_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

# Send alert if any usage exceeds threshold

if [ $(echo "$CPU\_USAGE > 80" | bc) -eq 1 ]; then

echo "High CPU usage detected: $CPU\_USAGE%" | mail -s "High CPU Usage Alert" admin@example.com

fi

if [ $(echo "$MEM\_USAGE > 80" | bc) -eq 1 ]; then

echo "High Memory usage detected: $MEM\_USAGE%" | mail -s "High Memory Usage Alert" admin@example.com

fi

if [ $DISK\_USAGE -gt 90 ]; then

echo "Disk space is running low: $DISK\_USAGE%" | mail -s "Disk Space Alert" admin@example.com

fi

1. **Automating Email Report Generation:**
   * Generate and send out automated daily, weekly, or monthly reports based on logs, system stats, or other metrics.

bash

CopyEdit

#!/bin/bash

# Generate system report and send it via email

REPORT\_FILE="/tmp/system\_report.txt"

echo "System Report: $(date)" > $REPORT\_FILE

echo "---------------------------" >> $REPORT\_FILE

echo "CPU Usage:" >> $REPORT\_FILE

top -bn1 | grep "Cpu(s)" >> $REPORT\_FILE

echo "Memory Usage:" >> $REPORT\_FILE

free >> $REPORT\_FILE

echo "Disk Usage:" >> $REPORT\_FILE

df -h >> $REPORT\_FILE

# Send report via email

mail -s "Daily System Report" user@example.com < $REPORT\_FILE

1. **Automating File Synchronization Between Servers:**
   * Use rsync to sync files from one server to another, ensuring that changes on the source server are reflected on the destination server.

bash

CopyEdit

#!/bin/bash

# Sync files between servers

SRC\_SERVER="user@source\_server:/path/to/files"

DEST\_SERVER="user@destination\_server:/path/to/destination"

rsync -avz $SRC\_SERVER $DEST\_SERVER

1. **Managing System Services (Start/Stop/Restart):**
   * Automate the management of system services like web servers, databases, etc.

bash

CopyEdit

#!/bin/bash

# Start, stop, or restart a service

SERVICE\_NAME="apache2"

case $1 in

start)

sudo systemctl start $SERVICE\_NAME

;;

stop)

sudo systemctl stop $SERVICE\_NAME

;;

restart)

sudo systemctl restart $SERVICE\_NAME

;;

status)

sudo systemctl status $SERVICE\_NAME

;;

\*)

echo "Usage: $0 {start|stop|restart|status}"

exit 1

esac

1. **Automating Data Migration (e.g., CSV to Database):**
   * Automate data migrations, such as importing CSV files into a database.

bash

CopyEdit

#!/bin/bash

# Import CSV data into MySQL

DB\_NAME="my\_database"

DB\_USER="root"

DB\_PASSWORD="password"

CSV\_FILE="/path/to/file.csv"

mysql -u $DB\_USER -p$DB\_PASSWORD $DB\_NAME -e "

LOAD DATA LOCAL INFILE '$CSV\_FILE'

INTO TABLE my\_table

FIELDS TERMINATED BY ','

ENCLOSED BY '\"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

"

1. **Automated SSL Certificate Expiry Monitoring:**
   * Monitor SSL certificates and send an alert when they are nearing expiry.

bash

CopyEdit

#!/bin/bash

# Check SSL certificate expiry

DOMAIN="example.com"

CERT\_EXPIRY\_DATE=$(echo | openssl s\_client -connect $DOMAIN:443 2>/dev/null | openssl x509 -noout -enddate)

EXPIRY\_DATE=$(echo $CERT\_EXPIRY\_DATE | sed 's/notAfter=//')

EXPIRY\_TIMESTAMP=$(date -d "$EXPIRY\_DATE" +%s)

CURRENT\_TIMESTAMP=$(date +%s)

DIFF=$((($EXPIRY\_TIMESTAMP - $CURRENT\_TIMESTAMP)/86400))

if [ $DIFF -le 30 ]; then

echo "SSL certificate for $DOMAIN is expiring in $DIFF days!" | mail -s "SSL Certificate Expiry Alert" admin@example.com

fi

These shell script tasks can help with automating administrative and maintenance tasks, system monitoring, service management, data processing, and more. They make daily operations smoother and reduce the risk of human error.

4o

**You said:**

continue with more advance tasks please

**ChatGPT said:**

Here are some more advanced shell script tasks that are valuable in real-world applications:

1. **Automating Cluster Node Health Check:**
   * In a distributed system or cluster, you may want to monitor the health of all nodes and send alerts if any of them are down.

bash

CopyEdit

#!/bin/bash

# Check if all nodes in a cluster are reachable

NODES=("node1.example.com" "node2.example.com" "node3.example.com")

for NODE in "${NODES[@]}"; do

ping -c 1 $NODE > /dev/null 2>&1

if [ $? -ne 0 ]; then

echo "Node $NODE is unreachable!" | mail -s "Cluster Node Down" admin@example.com

else

echo "Node $NODE is up"

fi

done

1. **Automating a Git Workflow for Continuous Integration (CI):**
   * Automate the process of pulling new code, running tests, and deploying code as part of a CI pipeline.

bash

CopyEdit

#!/bin/bash

# Automate CI workflow (git pull, test, deploy)

git pull origin main

npm install

npm test

if [ $? -eq 0 ]; then

echo "Tests passed, deploying..."

# Deploy to production (e.g., copy files, restart services)

rsync -avz /local/app/ user@prod-server:/var/www/app/

sudo systemctl restart myapp

else

echo "Tests failed, aborting deployment."

exit 1

fi

1. **Automating Multi-Server File Distribution:**
   * Distribute files to multiple servers, which is useful for updates or synchronization across different environments.

bash

CopyEdit

#!/bin/bash

# Distribute files to multiple servers

FILE="/path/to/file.txt"

SERVERS=("server1" "server2" "server3")

for SERVER in "${SERVERS[@]}"; do

scp $FILE user@$SERVER:/path/to/destination/

if [ $? -eq 0 ]; then

echo "File successfully copied to $SERVER"

else

echo "Failed to copy file to $SERVER" | mail -s "File Distribution Error" admin@example.com

fi

done

1. **Automating Server Provisioning with Cloud APIs (AWS, GCP, etc.):**
   * Automate the provisioning of new virtual machines or instances using cloud APIs such as AWS or GCP.

bash

CopyEdit

#!/bin/bash

# Provision a new AWS EC2 instance using AWS CLI

INSTANCE\_TYPE="t2.micro"

AMI\_ID="ami-xxxxxxxxxxxxxxx"

REGION="us-east-1"

KEY\_NAME="my-key-pair"

SECURITY\_GROUP="my-security-group"

INSTANCE\_ID=$(aws ec2 run-instances --image-id $AMI\_ID --instance-type $INSTANCE\_TYPE --key-name $KEY\_NAME --security-groups $SECURITY\_GROUP --region $REGION --query 'Instances[0].InstanceId' --output text)

echo "Provisioned new EC2 instance with ID $INSTANCE\_ID"

1. **Advanced Log Aggregation and Parsing:**
   * Aggregate and parse logs from multiple servers and generate a consolidated report for further analysis.

bash

CopyEdit

#!/bin/bash

# Aggregate logs from multiple servers and parse them

SERVERS=("server1" "server2" "server3")

LOG\_DIR="/var/log/myapp"

REPORT\_FILE="/tmp/aggregated\_log\_report.txt"

echo "Aggregated Logs:" > $REPORT\_FILE

for SERVER in "${SERVERS[@]}"; do

ssh user@$SERVER "cat $LOG\_DIR/\*.log" >> $REPORT\_FILE

done

# Parse logs for specific patterns (e.g., errors)

grep "ERROR" $REPORT\_FILE > /tmp/error\_log\_report.txt

1. **Automating Docker Container Management:**
   * Automate Docker container deployment, scaling, and monitoring.

bash

CopyEdit

#!/bin/bash

# Deploy a new Docker container and monitor its status

IMAGE\_NAME="myapp:latest"

CONTAINER\_NAME="myapp\_container"

docker run -d --name $CONTAINER\_NAME $IMAGE\_NAME

if [ $? -eq 0 ]; then

echo "Container $CONTAINER\_NAME started successfully"

else

echo "Failed to start container" | mail -s "Docker Deployment Error" admin@example.com

fi

# Monitor container status

docker ps -f "name=$CONTAINER\_NAME" --format "table {{.Names}}\t{{.Status}}"

1. **Automating SSH Key Management Across Multiple Servers:**
   * Distribute SSH keys to multiple servers, allowing passwordless login for automation purposes.

bash

CopyEdit

#!/bin/bash

# Deploy SSH public key to multiple servers for passwordless SSH

PUBLIC\_KEY="~/.ssh/id\_rsa.pub"

SERVERS=("server1" "server2" "server3")

for SERVER in "${SERVERS[@]}"; do

ssh-copy-id -i $PUBLIC\_KEY user@$SERVER

if [ $? -eq 0 ]; then

echo "SSH key deployed to $SERVER"

else

echo "Failed to deploy SSH key to $SERVER" | mail -s "SSH Key Deployment Failed" admin@example.com

fi

done

1. **Automating Database Replication Setup:**
   * Set up and automate database replication between master and slave databases.

bash

CopyEdit

#!/bin/bash

# Set up MySQL database replication

MASTER\_DB="master\_db"

SLAVE\_DB="slave\_db"

MASTER\_IP="192.168.1.1"

SLAVE\_IP="192.168.1.2"

# On master server

mysql -u root -p -e "GRANT REPLICATION SLAVE ON \*.\* TO 'replica\_user'@'%' IDENTIFIED BY 'password'; FLUSH PRIVILEGES;"

# On slave server

mysql -u root -p -e "CHANGE MASTER TO MASTER\_HOST='$MASTER\_IP', MASTER\_USER='replica\_user', MASTER\_PASSWORD='password', MASTER\_LOG\_FILE='mysql-bin.000001', MASTER\_LOG\_POS=154; START SLAVE;"

# Verify replication status

mysql -u root -p -e "SHOW SLAVE STATUS\G"

1. **Automating Scheduled Reports with Data Aggregation:**
   * Automate the collection, aggregation, and reporting of system data like server performance, traffic statistics, or application usage.

bash

CopyEdit

#!/bin/bash

# Generate weekly system report with aggregated data

REPORT\_FILE="/tmp/system\_report\_$(date +'%Y-%m-%d').txt"

echo "Weekly System Report: $(date)" > $REPORT\_FILE

echo "CPU Usage:" >> $REPORT\_FILE

top -bn1 | grep "Cpu(s)" >> $REPORT\_FILE

echo "Disk Usage:" >> $REPORT\_FILE

df -h >> $REPORT\_FILE

echo "Memory Usage:" >> $REPORT\_FILE

free -m >> $REPORT\_FILE

# Send the report via email

mail -s "Weekly System Report" admin@example.com < $REPORT\_FILE

1. **Automating Cluster Scaling with Kubernetes:**
   * Automate the scaling of a Kubernetes cluster based on CPU and memory usage thresholds.

bash

CopyEdit

#!/bin/bash

# Check resource usage in a Kubernetes cluster and scale up/down

CPU\_USAGE=$(kubectl top nodes | awk '{if(NR>1)print $3}' | sed 's/%//g' | sort -n | tail -n 1)

MEMORY\_USAGE=$(kubectl top nodes | awk '{if(NR>1)print $4}' | sed 's/Mi//g' | sort -n | tail -n 1)

if [ $CPU\_USAGE -gt 80 ]; then

kubectl scale deployment myapp --replicas=5

echo "Scaling up myapp deployment"

fi

if [ $MEMORY\_USAGE -gt 75 ]; then

kubectl scale deployment myapp --replicas=2

echo "Scaling down myapp deployment"

fi

These advanced shell script tasks involve managing complex systems and automating crucial processes such as cloud provisioning, container management, CI/CD pipelines, database replication, cluster scaling, and more. These are ideal for handling enterprise-level tasks, improving efficiency, and maintaining the reliability of large systems and infrastructures.

4o

**You said:**

continue with more advanced tasks

**ChatGPT said:**

Certainly! Here are even more advanced shell script tasks that can be highly beneficial for automating complex and large-scale operations:

1. **Automating Infrastructure as Code (IaC) with Terraform:**
   * You can automate the process of provisioning and managing cloud infrastructure using Terraform.

bash

CopyEdit

#!/bin/bash

# Automate Terraform Infrastructure Provisioning

TERRAFORM\_DIR="/path/to/terraform/configuration"

cd $TERRAFORM\_DIR

terraform init

terraform plan -out=tfplan

terraform apply tfplan

# Capture the output for logging or reporting

terraform output > terraform\_output.log

echo "Terraform provisioning completed." | mail -s "Infrastructure Provisioning Report" admin@example.com

1. **Automating Cross-Server Synchronization with Rsync and SSH:**
   * For large-scale deployments, you can synchronize files across multiple servers in a more scalable way with error handling and logging.

bash

CopyEdit

#!/bin/bash

# Synchronize files to multiple servers with Rsync

SRC\_DIR="/path/to/source/directory/"

SERVERS=("server1" "server2" "server3")

LOG\_FILE="/var/log/rsync\_sync.log"

for SERVER in "${SERVERS[@]}"; do

rsync -avz --delete $SRC\_DIR user@$SERVER:/path/to/destination/ >> $LOG\_FILE 2>&1

if [ $? -eq 0 ]; then

echo "Sync to $SERVER completed successfully" >> $LOG\_FILE

else

echo "Sync to $SERVER failed" | mail -s "Rsync Error on $SERVER" admin@example.com

fi

done

1. **Automating Security Patching and Updates for Linux Servers:**
   * Automatically patch and update security packages, then verify the system integrity.

bash

CopyEdit

#!/bin/bash

# Automate security patching and updates

echo "Starting system update..."

sudo apt-get update -y

sudo apt-get upgrade -y

sudo apt-get dist-upgrade -y

sudo apt-get autoremove -y

# Check for security patches

sudo unattended-upgrade -d

# Verify system integrity (optional)

sudo debsums -s

echo "System updated successfully" | mail -s "Security Patching Complete" admin@example.com

1. **Automating High Availability Setup with HAProxy:**
   * Automatically configure HAProxy for load balancing and high availability.

bash

CopyEdit

#!/bin/bash

# Configure HAProxy for load balancing

BACKEND\_SERVERS=("192.168.1.2:80" "192.168.1.3:80")

FRONTEND\_PORT=80

CONFIG\_FILE="/etc/haproxy/haproxy.cfg"

echo "global

log 127.0.0.1 local0

maxconn 200

defaults

log global

option httplog

timeout connect 5000ms

timeout client 50000ms

timeout server 50000ms

frontend http\_front

bind \*:$FRONTEND\_PORT

default\_backend http\_back

backend http\_back" > $CONFIG\_FILE

for SERVER in "${BACKEND\_SERVERS[@]}"; do

echo " server $SERVER check" >> $CONFIG\_FILE

done

systemctl restart haproxy

1. **Automating Server Cloning (VMs or Containers):**
   * Clone a server or virtual machine using shell scripts, useful in large environments where rapid scaling is necessary.

bash

CopyEdit

#!/bin/bash

# Clone a VM (using KVM as an example)

ORIGINAL\_VM="original\_vm\_name"

NEW\_VM="cloned\_vm\_name"

TEMPLATE\_DIR="/var/lib/libvirt/images"

TEMPLATE\_PATH="$TEMPLATE\_DIR/$ORIGINAL\_VM.qcow2"

NEW\_VM\_PATH="$TEMPLATE\_DIR/$NEW\_VM.qcow2"

cp $TEMPLATE\_PATH $NEW\_VM\_PATH

virsh define /etc/libvirt/qemu/$NEW\_VM.xml

virsh start $NEW\_VM

echo "Cloning completed: $NEW\_VM" | mail -s "VM Cloning Report" admin@example.com

1. **Automating Network Monitoring and Bandwidth Alerts:**
   * Automate network monitoring to check for bandwidth usage or abnormal traffic patterns and send alerts.

bash

CopyEdit

#!/bin/bash

# Monitor network traffic and alert on high usage

INTERFACE="eth0"

THRESHOLD=80

BANDWIDTH\_USAGE=$(ifstat -i $INTERFACE 1 1 | awk 'NR==3 {print $1}')

if [ $(echo "$BANDWIDTH\_USAGE > $THRESHOLD" | bc) -eq 1 ]; then

echo "High network usage detected: $BANDWIDTH\_USAGE Mbps" | mail -s "Network Bandwidth Alert" admin@example.com

fi

1. **Automating Kubernetes Cluster Setup and Node Registration:**
   * Automate the setup of a Kubernetes cluster and the addition of worker nodes.

bash

CopyEdit

#!/bin/bash

# Setup Kubernetes master node

KUBE\_VERSION="1.21.0"

MASTER\_NODE\_IP="192.168.1.100"

# Install Kubernetes components

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

sudo apt-add-repository "deb https://apt.kubernetes.io/ kubernetes-xenial main"

sudo apt-get update -y

sudo apt-get install -y kubeadm=$KUBE\_VERSION-00 kubelet=$KUBE\_VERSION-00 kubectl=$KUBE\_VERSION-00

# Initialize Kubernetes cluster

sudo kubeadm init --apiserver-advertise-address=$MASTER\_NODE\_IP --pod-network-cidr=10.244.0.0/16

# Set up kubeconfig

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

# Install Calico network plugin

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

1. **Automating Service Discovery and Configuration Management:**
   * Automate the management of service discovery using tools like Consul, etcd, or Zookeeper to keep track of services in a microservices architecture.

bash

CopyEdit

#!/bin/bash

# Register a service with Consul for service discovery

SERVICE\_NAME="myapp"

SERVICE\_ID="myapp\_$(hostname)"

SERVICE\_PORT=8080

curl -X PUT -d '{"ID": "'$SERVICE\_ID'", "Name": "'$SERVICE\_NAME'", "Port": '$SERVICE\_PORT'}' \

http://localhost:8500/v1/agent/service/register

echo "Service $SERVICE\_NAME registered with Consul"

1. **Automating Multi-Region Cloud Setup:**
   * Automate the creation of cloud resources in multiple regions across providers like AWS, GCP, or Azure for redundancy and disaster recovery.

bash

CopyEdit

#!/bin/bash

# Automate resource provisioning in multiple cloud regions (AWS example)

REGIONS=("us-east-1" "us-west-1" "eu-west-1")

INSTANCE\_TYPE="t2.micro"

IMAGE\_ID="ami-xxxxxxxxxxxxxxx"

for REGION in "${REGIONS[@]}"; do

aws ec2 run-instances --image-id $IMAGE\_ID --instance-type $INSTANCE\_TYPE --region $REGION

if [ $? -eq 0 ]; then

echo "Instance launched in $REGION"

else

echo "Failed to launch instance in $REGION" | mail -s "Cloud Provisioning Error in $REGION" admin@example.com

fi

done

1. **Automating Load Testing for Web Applications:**
   * Run automated load testing against web applications using tools like ab (Apache Benchmark) or siege for performance analysis.

bash

CopyEdit

#!/bin/bash

# Run load testing on a web application

URL="http://example.com"

CONCURRENCY=50

REQUESTS=1000

ab -n $REQUESTS -c $CONCURRENCY $URL > /tmp/load\_test\_results.txt

cat /tmp/load\_test\_results.txt | mail -s "Load Test Results for $URL" admin@example.com

1. **Automating Database Sharding Setup:**
   * Automate the setup of database sharding to scale large databases horizontally.

bash

CopyEdit

#!/bin/bash

# Set up database sharding (MySQL example)

MASTER\_DB="main\_db"

SHARD\_DB1="shard1\_db"

SHARD\_DB2="shard2\_db"

MASTER\_USER="root"

MASTER\_PASSWORD="password"

# Create databases

mysql -u $MASTER\_USER -p$MASTER\_PASSWORD -e "CREATE DATABASE $SHARD\_DB1;"

mysql -u $MASTER\_USER -p$MASTER\_PASSWORD -e "CREATE DATABASE $SHARD\_DB2;"

# Set up replication (simplified example)

mysql -u $MASTER\_USER -p$MASTER\_PASSWORD -e "CHANGE MASTER TO MASTER\_HOST='master\_host', MASTER\_USER='replica\_user', MASTER\_PASSWORD='replica\_password';"

These advanced tasks focus on large-scale infrastructure management, high availability, cloud resource provisioning, service discovery, automation of security patching, and database scaling. They are invaluable in enterprise-level systems and cloud environments, making them critical for maintaining uptime, reducing manual work, and ensuring system scalability.

4o

**You said:**

continue with moderate tasks please a

**ChatGPT said:**

Certainly! Here are some **moderate-level** shell script tasks that can be highly practical in real-world scenarios:

**43. Automating File System Monitoring:**

* Set up file system monitoring to alert you when a specific directory exceeds a certain size, which is useful for keeping track of log files or data storage.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor directory size and alert if it exceeds the threshold

DIR="/var/log/myapp"

THRESHOLD=1000 # 1 GB

DIR\_SIZE=$(du -s $DIR | awk '{print $1}')

if [ $DIR\_SIZE -gt $THRESHOLD ]; then

echo "Warning: Directory $DIR size exceeds threshold ($DIR\_SIZE KB)" | mail -s "Disk Usage Alert" admin@example.com

fi

```

**44. Automating Backup of Database with Compression:**

* Automatically back up a MySQL database, compress it, and store it with a timestamp.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# MySQL database backup with compression

DB\_NAME="my\_database"

DB\_USER="root"

DB\_PASSWORD="password"

BACKUP\_DIR="/backup/mysql"

DATE=$(date +'%Y%m%d%H%M%S')

mysqldump -u $DB\_USER -p$DB\_PASSWORD $DB\_NAME | gzip > $BACKUP\_DIR/db\_backup\_$DATE.sql.gz

echo "Backup completed for $DB\_NAME at $DATE" | mail -s "Database Backup Completed" admin@example.com

```

**45. Automating Log Aggregation from Multiple Servers:**

* Aggregate logs from multiple servers and combine them into a central log file for analysis.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Aggregate logs from multiple servers

SERVERS=("server1" "server2" "server3")

LOG\_DIR="/var/log"

AGGREGATED\_LOG="/tmp/aggregated\_logs.txt"

for SERVER in "${SERVERS[@]}"; do

scp user@$SERVER:$LOG\_DIR/\*.log >> $AGGREGATED\_LOG

done

echo "Log aggregation completed" | mail -s "Log Aggregation Report" admin@example.com

```

**46. Automating Web Scraping (using curl and grep):**

* Automate the scraping of specific information from a website, such as extracting specific text or metadata.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Scrape website for title tag and save to file

URL="https://example.com"

TITLE=$(curl -s $URL | grep -oP '(?<=<title>)(.\*)(?=</title>)')

echo "Page title for $URL: $TITLE" > /tmp/scraped\_title.txt

```

**47. Automating File Backup with Incremental Changes:**

* Backup only the files that have changed since the last backup using rsync.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Perform incremental backup using rsync

SOURCE\_DIR="/home/user/data"

BACKUP\_DIR="/backup/data"

rsync -av --link-dest=$BACKUP\_DIR/latest/ $SOURCE\_DIR $BACKUP\_DIR/$(date +'%Y%m%d%H%M%S')

# Update the 'latest' symlink

rm -f $BACKUP\_DIR/latest

ln -s $BACKUP\_DIR/$(date +'%Y%m%d%H%M%S') $BACKUP\_DIR/latest

echo "Incremental backup completed." | mail -s "Incremental Backup Report" admin@example.com

```

**48. Automating System Package Updates:**

* Automate regular system updates, including security patches.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Automatically update the system and install security patches

sudo apt-get update -y

sudo apt-get upgrade -y

sudo apt-get dist-upgrade -y

sudo apt-get autoremove -y

sudo unattended-upgrade -d

echo "System updated successfully" | mail -s "System Update Completed" admin@example.com

```

**49. Automating File Permissions Management:**

* Automatically set permissions for files and directories, ensuring proper access control.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Change file permissions for a directory

DIR="/var/www/myapp"

USER="www-data"

GROUP="www-data"

# Set ownership and permissions

chown -R $USER:$GROUP $DIR

chmod -R 755 $DIR

echo "Permissions set for $DIR" | mail -s "File Permissions Update" admin@example.com

```

**50. Automating System Cleanup (Unnecessary Files):**

* Automatically clean up unused packages, cache, and temporary files to free up disk space.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Clean up unnecessary files and clear package cache

sudo apt-get autoremove -y

sudo apt-get clean -y

sudo rm -rf /tmp/\*

sudo rm -rf ~/.cache/\*

echo "System cleanup completed" | mail -s "System Cleanup Report" admin@example.com

```

**51. Automating Email Backup with Gmail (using msmtp):**

* Automate the process of backing up emails using a command-line email client like msmtp and a simple cron job.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Backup Gmail emails using msmtp

EMAIL="your-email@gmail.com"

BACKUP\_DIR="/backup/emails"

DATE=$(date +'%Y%m%d')

msmtp --from=$EMAIL --logfile=$BACKUP\_DIR/msmtp\_log\_$DATE.txt < /tmp/email\_backup\_request.txt

echo "Email backup completed for $EMAIL at $DATE" | mail -s "Email Backup Report" admin@example.com

```

**52. Automating SSL Certificate Renewal with Let’s Encrypt:**

* Automate the renewal of SSL certificates using Certbot for your web servers.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Renew SSL certificate with Certbot

sudo certbot renew --quiet

sudo systemctl reload apache2 # or nginx

echo "SSL certificate renewed and web server reloaded" | mail -s "SSL Certificate Renewal" admin@example.com

```

**53. Automating Process Monitoring and Restart:**

* Automatically monitor and restart services or processes that have crashed or stopped.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor a service and restart it if it's not running

SERVICE\_NAME="apache2"

if ! systemctl is-active --quiet $SERVICE\_NAME; then

echo "$SERVICE\_NAME is down! Restarting..." | mail -s "$SERVICE\_NAME Down Alert" admin@example.com

sudo systemctl restart $SERVICE\_NAME

fi

```

**54. Automating Disk Usage Report Generation:**

* Automatically generate disk usage reports and email them periodically.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Generate disk usage report

REPORT\_FILE="/tmp/disk\_usage\_report.txt"

df -h > $REPORT\_FILE

du -sh /var/log >> $REPORT\_FILE

du -sh /home >> $REPORT\_FILE

cat $REPORT\_FILE | mail -s "Disk Usage Report" admin@example.com

```

**55. Automating Disk Encryption Backup (LUKS):**

* Create a backup script for disk encryption (e.g., using LUKS) and save the encryption keys securely.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Backup encrypted disk image and encryption key

SOURCE\_DIR="/path/to/encrypted\_disk"

BACKUP\_DIR="/backup/encrypted\_disk"

KEY\_FILE="/path/to/keyfile"

# Backup encrypted volume

dd if=$SOURCE\_DIR of=$BACKUP\_DIR/encrypted\_disk\_backup.img bs=64K

# Backup encryption key

cp $KEY\_FILE $BACKUP\_DIR/keyfile\_backup

echo "Encrypted disk backup completed." | mail -s "Disk Encryption Backup Report" admin@example.com

```

**56. Automating Server Health Checks (CPU, Memory, Disk Usage):**

* Monitor the health of your server and send a report if any resource is above the set threshold.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor CPU, memory, and disk usage and send an alert if above threshold

CPU\_THRESHOLD=85

MEM\_THRESHOLD=80

DISK\_THRESHOLD=90

CPU\_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.\*, \*\([0-9.]\*\)%\* id.\*/\1/" | awk '{print 100 - $1}')

MEM\_USAGE=$(free | grep Mem | awk '{print $3/$2 \* 100.0}')

DISK\_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

if [ $(echo "$CPU\_USAGE > $CPU\_THRESHOLD" | bc) -eq 1 ]; then

echo "High CPU usage: $CPU\_USAGE%" | mail -s "High CPU Usage Alert" admin@example.com

fi

if [ $(echo "$MEM\_USAGE > $MEM\_THRESHOLD" | bc) -eq 1 ]; then

echo "High Memory usage: $MEM\_USAGE%" | mail -s "High Memory Usage Alert" admin@example.com

fi

if [ $DISK\_USAGE -gt $DISK\_THRESHOLD ]; then

echo "Disk usage is above threshold: $DISK\_USAGE%" | mail -s "Disk Usage Alert" admin@example.com

fi

```

**57. Automating Simple File Synchronization with rsync:**

* Synchronize files between local and remote machines for backups or data replication.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Synchronize files using rsync

SRC\_DIR="/path/to/source"

DEST\_DIR="user@remote:/path/to/destination"

rsync -avz $SRC\_DIR $DEST\_DIR

echo "File synchronization completed" | mail -s "File Sync Report" admin@example.com

```

These tasks can help automate everyday administrative tasks, improve system performance, manage backups, monitor system health, and provide essential security features. They strike a balance between simplicity and functionality, making them highly useful for system administrators and developers.

4o

**You said:**

continue please

**ChatGPT said:**

Here are more **moderate-level** shell script tasks that can be quite useful in a variety of real-world scenarios:

**58. Automating User Account Management:**

* Automate the process of adding, modifying, or deleting user accounts.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Manage user accounts (add, modify, delete)

ACTION=$1

USERNAME=$2

case $ACTION in

add)

sudo useradd -m $USERNAME

sudo passwd $USERNAME

echo "User $USERNAME added successfully." | mail -s "User Account Created" admin@example.com

;;

modify)

sudo usermod -aG sudo $USERNAME

echo "User $USERNAME added to sudo group." | mail -s "User Group Modified" admin@example.com

;;

delete)

sudo userdel -r $USERNAME

echo "User $USERNAME deleted." | mail -s "User Account Deleted" admin@example.com

;;

\*)

echo "Usage: $0 {add|modify|delete} username"

exit 1

;;

esac

```

**59. Automating Database Backup and Restoration:**

* Automate the backup of a MySQL database and restoration from backup.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Backup MySQL database and restore from backup

DB\_NAME="my\_database"

DB\_USER="root"

DB\_PASSWORD="password"

BACKUP\_DIR="/backup/mysql"

BACKUP\_FILE="$BACKUP\_DIR/db\_backup\_$(date +'%Y%m%d').sql"

# Backup the database

mysqldump -u $DB\_USER -p$DB\_PASSWORD $DB\_NAME > $BACKUP\_FILE

echo "Backup completed for $DB\_NAME" | mail -s "MySQL Database Backup" admin@example.com

# Restore the database from a backup

# mysql -u $DB\_USER -p$DB\_PASSWORD $DB\_NAME < $BACKUP\_FILE

```

**60. Automating Firewall Configuration (using ufw):**

* Automate the process of configuring firewall rules to allow or block certain ports.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Configure UFW firewall rules

ACTION=$1

PORT=$2

case $ACTION in

allow)

sudo ufw allow $PORT

echo "Port $PORT allowed through firewall" | mail -s "UFW Firewall Configuration" admin@example.com

;;

deny)

sudo ufw deny $PORT

echo "Port $PORT denied through firewall" | mail -s "UFW Firewall Configuration" admin@example.com

;;

\*)

echo "Usage: $0 {allow|deny} port\_number"

exit 1

;;

esac

```

**61. Automating Service Monitoring with systemctl:**

* Monitor system services and restart them if they are down.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor system services and restart if not running

SERVICE\_NAME="apache2"

systemctl is-active --quiet $SERVICE\_NAME

if [ $? -ne 0 ]; then

echo "$SERVICE\_NAME is down. Restarting..." | mail -s "$SERVICE\_NAME Down Alert" admin@example.com

sudo systemctl restart $SERVICE\_NAME

else

echo "$SERVICE\_NAME is running."

fi

```

**62. Automating Disk Space Alerts:**

* Monitor the disk space usage and send alerts when disk space exceeds a certain threshold.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor disk usage and alert if it exceeds a threshold

THRESHOLD=80 # 80% threshold

DISK\_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

if [ $DISK\_USAGE -gt $THRESHOLD ]; then

echo "Disk usage is above threshold: $DISK\_USAGE%" | mail -s "Disk Usage Alert" admin@example.com

fi

```

**63. Automating Log Rotation:**

* Automate the log rotation to prevent log files from growing too large.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Rotate logs for a specific application

LOG\_DIR="/var/log/myapp"

LOG\_FILE="myapp.log"

BACKUP\_DIR="/backup/logs"

TIMESTAMP=$(date +'%Y%m%d')

# Rotate the log file

mv $LOG\_DIR/$LOG\_FILE $BACKUP\_DIR/$LOG\_FILE.$TIMESTAMP

touch $LOG\_DIR/$LOG\_FILE

gzip $BACKUP\_DIR/$LOG\_FILE.$TIMESTAMP

echo "Log rotation completed" | mail -s "Log Rotation Report" admin@example.com

```

**64. Automating File Synchronization Across Multiple Servers:**

* Use rsync to synchronize files between a central server and multiple remote servers.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Synchronize files to multiple servers

SRC\_DIR="/path/to/source"

DEST\_DIR="user@server:/path/to/destination"

SERVERS=("server1" "server2" "server3")

for SERVER in "${SERVERS[@]}"; do

rsync -avz $SRC\_DIR $SERVER:$DEST\_DIR

if [ $? -eq 0 ]; then

echo "Files synced to $SERVER successfully."

else

echo "Error syncing files to $SERVER" | mail -s "Rsync Sync Failure" admin@example.com

fi

done

```

**65. Automating Git Repository Backup:**

* Automatically back up a Git repository to a remote server.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Back up a local Git repository to a remote server

REPO\_DIR="/path/to/repository"

REMOTE\_SERVER="user@server:/path/to/backup"

BACKUP\_DIR="$REMOTE\_SERVER/$(date +'%Y%m%d')"

# Create backup directory on remote server

ssh user@server "mkdir -p $BACKUP\_DIR"

# Backup the repository

rsync -avz $REPO\_DIR $BACKUP\_DIR

echo "Git repository backed up to $BACKUP\_DIR" | mail -s "Git Repository Backup" admin@example.com

```

**66. Automating File Archiving with Compression:**

* Automatically archive and compress files into a .tar.gz format.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Archive and compress files

SOURCE\_DIR="/path/to/directory"

ARCHIVE\_NAME="backup\_$(date +'%Y%m%d').tar.gz"

DEST\_DIR="/path/to/backup"

tar -czf $DEST\_DIR/$ARCHIVE\_NAME -C $SOURCE\_DIR .

echo "Backup archived and compressed into $ARCHIVE\_NAME" | mail -s "File Archive Completed" admin@example.com

```

**67. Automating Server Reboot Schedule:**

* Schedule a server reboot for regular maintenance and ensure it runs during off-peak hours.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Schedule a reboot for the server at 3:00 AM

sudo shutdown -r 03:00

echo "Server reboot scheduled for 3:00 AM." | mail -s "Scheduled Server Reboot" admin@example.com

```

**68. Automating Process Monitoring with Alerts:**

* Monitor the status of critical processes (e.g., web server) and alert if they are down.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor critical process (e.g., apache2)

PROCESS\_NAME="apache2"

if ! pgrep -x "$PROCESS\_NAME" > /dev/null; then

echo "$PROCESS\_NAME is not running!" | mail -s "$PROCESS\_NAME Process Alert" admin@example.com

else

echo "$PROCESS\_NAME is running."

fi

```

**69. Automating Cleanup of Old Log Files:**

* Automatically remove old log files that are no longer needed to free up disk space.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Remove log files older than 30 days

LOG\_DIR="/var/log/myapp"

find $LOG\_DIR -name "\*.log" -type f -mtime +30 -exec rm -f {} \;

echo "Old log files removed from $LOG\_DIR" | mail -s "Log Cleanup Completed" admin@example.com

```

**70. Automating S3 Bucket Backup (AWS CLI):**

* Use AWS CLI to back up a directory to an S3 bucket.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Backup directory to AWS S3 bucket

DIR\_TO\_BACKUP="/path/to/directory"

S3\_BUCKET="s3://my-backup-bucket"

aws s3 sync $DIR\_TO\_BACKUP $S3\_BUCKET

echo "Backup completed to S3 bucket $S3\_BUCKET" | mail -s "S3 Backup Report" admin@example.com

```

**71. Automating Regular Data Cleanup:**

* Automate the removal of old data files that are no longer needed.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Clean up old data files (older than 60 days)

DATA\_DIR="/path/to/data"

find $DATA\_DIR -type f -mtime +60 -exec rm -f {} \;

echo "Old data files cleaned up from $DATA\_DIR" | mail -s "Data Cleanup Report" admin@example.com

```

These tasks are designed to be simple to implement while still being powerful tools for system administrators or DevOps engineers. Automating routine tasks such as backups, monitoring, and cleanup can help ensure the smooth operation of systems while reducing the amount of manual intervention required.

4o

**You said:**

continue please with more real worlds tasks please

**ChatGPT said:**

Here are even **more real-world shell script tasks** that are practical, useful, and commonly applied in a variety of environments:

**72. Automating SSL Certificate Backup:**

* Automatically back up SSL certificates and their associated private keys to a secure location.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Backup SSL certificates and private keys

SSL\_CERT\_DIR="/etc/ssl/certs"

SSL\_KEY\_DIR="/etc/ssl/private"

BACKUP\_DIR="/backup/ssl"

# Backup certificates and keys

cp -r $SSL\_CERT\_DIR $BACKUP\_DIR/certs

cp -r $SSL\_KEY\_DIR $BACKUP\_DIR/private

echo "SSL certificate and key backup completed." | mail -s "SSL Backup Report" admin@example.com

```

**73. Automating Application Deployment with Git:**

* Automate the deployment process by pulling the latest code from a Git repository and restarting the web server.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Deploy latest code from Git and restart web server

APP\_DIR="/var/www/myapp"

GIT\_REPO="https://github.com/username/myapp.git"

SERVICE\_NAME="apache2"

# Pull latest code

cd $APP\_DIR

git pull $GIT\_REPO

# Restart the web server

systemctl restart $SERVICE\_NAME

echo "Deployment completed and $SERVICE\_NAME restarted" | mail -s "Deployment Report" admin@example.com

```

**74. Automating API Call to External Service:**

* Automate a simple API request and save the result to a log file or database.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Call an API and save the result

API\_URL="https://api.example.com/data"

RESPONSE=$(curl -s $API\_URL)

LOG\_FILE="/var/log/api\_response.log"

echo "Response from API at $(date): $RESPONSE" >> $LOG\_FILE

echo "API call completed and logged." | mail -s "API Call Report" admin@example.com

```

**75. Automating File Integrity Check:**

* Check for file integrity using checksums and notify if files have been modified or tampered with.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Check file integrity using checksums

FILE="/path/to/important/file"

CHECKSUM\_FILE="/path/to/checksum\_file.sha256"

CURRENT\_CHECKSUM=$(sha256sum $FILE | awk '{print $1}')

STORED\_CHECKSUM=$(cat $CHECKSUM\_FILE)

if [ "$CURRENT\_CHECKSUM" != "$STORED\_CHECKSUM" ]; then

echo "File integrity compromised: $FILE" | mail -s "File Integrity Alert" admin@example.com

else

echo "File integrity is intact."

fi

```

**76. Automating Server Inventory Management:**

* Generate and save an inventory of all active servers in your network (including IP, OS, etc.) for auditing.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Generate server inventory

SERVER\_LIST=("server1" "server2" "server3")

INVENTORY\_FILE="/tmp/server\_inventory.txt"

for SERVER in "${SERVER\_LIST[@]}"; do

OS=$(ssh $SERVER "uname -a")

IP=$(ssh $SERVER "hostname -I")

echo "$SERVER - OS: $OS, IP: $IP" >> $INVENTORY\_FILE

done

echo "Server inventory generated and saved to $INVENTORY\_FILE" | mail -s "Server Inventory Report" admin@example.com

```

**77. Automating Regular MySQL Data Import:**

* Automate the import of a new dataset into a MySQL database from a CSV file.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Import CSV data into MySQL

DB\_NAME="my\_database"

DB\_USER="root"

DB\_PASSWORD="password"

CSV\_FILE="/path/to/data.csv"

mysql -u $DB\_USER -p$DB\_PASSWORD $DB\_NAME -e "

LOAD DATA LOCAL INFILE '$CSV\_FILE'

INTO TABLE my\_table

FIELDS TERMINATED BY ','

ENCLOSED BY '\"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

"

echo "Data import completed from $CSV\_FILE" | mail -s "Data Import Report" admin@example.com

```

**78. Automating Security Scanning for Vulnerabilities:**

* Automate the process of running a security scan for known vulnerabilities and send an alert if issues are found.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Run security scan (e.g., with OpenVAS or another scanner)

SCAN\_RESULT=$(openvas-check --host localhost --port 9390)

if echo "$SCAN\_RESULT" | grep -q "vulnerabilities found"; then

echo "Security scan found vulnerabilities!" | mail -s "Security Scan Alert" admin@example.com

else

echo "Security scan completed with no issues."

fi

```

**79. Automating VM Snapshots (Using virsh):**

* Automatically take snapshots of a virtual machine running on KVM (or other virtual machines).

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Take a snapshot of a VM

VM\_NAME="my\_vm"

SNAPSHOT\_NAME="snapshot\_$(date +'%Y%m%d%H%M%S')"

virsh snapshot-create-as --domain $VM\_NAME --name $SNAPSHOT\_NAME --description "Automated snapshot"

echo "Snapshot $SNAPSHOT\_NAME created for $VM\_NAME." | mail -s "VM Snapshot Report" admin@example.com

```

**80. Automating Database Index Optimization:**

* Run automatic index optimization for your MySQL database to improve query performance.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Optimize database indexes for better performance

DB\_NAME="my\_database"

DB\_USER="root"

DB\_PASSWORD="password"

mysql -u $DB\_USER -p$DB\_PASSWORD -e "

USE $DB\_NAME;

OPTIMIZE TABLE my\_table;

"

echo "Database index optimization completed for $DB\_NAME." | mail -s "Database Optimization Report" admin@example.com

```

**81. Automating System Performance Monitoring:**

* Monitor the performance of critical system resources like CPU, memory, and disk usage and send periodic reports.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor system resources

REPORT\_FILE="/tmp/system\_performance\_report.txt"

echo "System Performance Report - $(date)" > $REPORT\_FILE

echo "-----------------------------" >> $REPORT\_FILE

echo "CPU Usage:" >> $REPORT\_FILE

top -bn1 | grep "Cpu(s)" >> $REPORT\_FILE

echo "Memory Usage:" >> $REPORT\_FILE

free -m >> $REPORT\_FILE

echo "Disk Usage:" >> $REPORT\_FILE

df -h >> $REPORT\_FILE

cat $REPORT\_FILE | mail -s "System Performance Report" admin@example.com

```

**82. Automating Software License Check:**

* Automatically check software license expiration dates and notify the admin before expiration.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Check software license expiration

LICENSE\_FILE="/path/to/license.txt"

EXPIRATION\_DATE=$(grep "Expiration" $LICENSE\_FILE | cut -d' ' -f2)

CURRENT\_DATE=$(date +%Y-%m-%d)

if [[ "$EXPIRATION\_DATE" < "$CURRENT\_DATE" ]]; then

echo "License expired! Please renew." | mail -s "License Expiration Alert" admin@example.com

else

echo "License is valid until $EXPIRATION\_DATE."

fi

```

**83. Automating Time Synchronization Across Servers:**

* Ensure that time across all servers is synchronized using NTP.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Synchronize time using NTP on all servers

SERVERS=("server1" "server2" "server3")

for SERVER in "${SERVERS[@]}"; do

ssh user@$SERVER "sudo systemctl restart ntp"

if [ $? -eq 0 ]; then

echo "Time synchronized on $SERVER"

else

echo "Failed to synchronize time on $SERVER" | mail -s "Time Sync Failure" admin@example.com

fi

done

```

**84. Automating Regular Data Migration Between Servers:**

* Automatically migrate data from one server to another, useful in server migrations or data redundancy setups.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Migrate data between two servers

SRC\_SERVER="user@source-server:/path/to/data"

DEST\_SERVER="user@destination-server:/path/to/destination"

rsync -avz $SRC\_SERVER $DEST\_SERVER

if [ $? -eq 0 ]; then

echo "Data migration completed successfully." | mail -s "Data Migration Report" admin@example.com

else

echo "Data migration failed." | mail -s "Data Migration Error" admin@example.com

fi

```

**85. Automating AWS EC2 Instance Health Check:**

* Monitor the health of EC2 instances and take actions (like restarting or terminating) if they are not healthy.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor EC2 instance health status

INSTANCE\_ID="i-xxxxxxxxxxxxxxxxx"

STATUS=$(aws ec2 describe-instance-status --instance-id $INSTANCE\_ID --query 'InstanceStatuses[0].InstanceState.Name' --output text)

if [ "$STATUS" != "running" ]; then

echo "EC2 instance $INSTANCE\_ID is down! Taking action..." | mail -s "EC2 Instance Health Alert" admin@example.com

aws ec2 reboot-instances --instance-ids $INSTANCE\_ID

else

echo "EC2 instance $INSTANCE\_ID is running smoothly."

fi

```

**86. Automating Scheduled Task Execution (Cron Jobs):**

* Schedule a task (such as a backup or maintenance script) to run automatically at a set time using cron.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Schedule a task using cron

CRON\_JOB="0 2 \* \* \* /path/to/script.sh"

(crontab -l; echo "$CRON\_JOB") | crontab -

echo "Cron job scheduled to run at 2:00 AM daily." | mail -s "Cron Job Scheduled" admin@example.com

```

These **real-world tasks** focus on key areas like security, data migration, system maintenance, performance monitoring, and more, helping to automate and streamline various administrative and operational activities. They can help save time, reduce human error, and ensure that your systems are consistently running smoothly.

4o

**You said:**

continue please with more real world tasks that are used in real world

**ChatGPT said:**

Here are **more real-world shell script tasks** that can be extremely useful in various scenarios:

**87. Automating Cloud Storage Backup:**

* Automatically back up files or directories to a cloud storage provider like Google Cloud Storage or AWS S3.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Backup local files to Google Cloud Storage

LOCAL\_DIR="/path/to/local/backup"

BUCKET\_NAME="gs://my-backup-bucket"

# Sync files to cloud

gsutil rsync -r $LOCAL\_DIR $BUCKET\_NAME

echo "Backup to Google Cloud Storage completed." | mail -s "Cloud Storage Backup Report" admin@example.com

```

**88. Automating Cron Jobs with Randomized Delay:**

* Add a randomized delay to cron jobs to avoid resource spikes from simultaneous execution.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Add randomized delay to cron job execution

DELAY=$((RANDOM % 60)) # Random delay between 0 and 59 seconds

(sleep $DELAY && /path/to/your/script.sh) &

echo "Cron job added with a randomized delay of $DELAY seconds." | mail -s "Cron Job with Random Delay" admin@example.com

```

**89. Automating Server Hardening:**

* Automate the process of applying security patches, configuring firewall, and disabling unused services.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Automate server hardening

# Update and upgrade system

sudo apt-get update -y

sudo apt-get upgrade -y

# Install UFW and configure firewall

sudo ufw enable

sudo ufw allow ssh

sudo ufw allow http

sudo ufw allow https

sudo ufw deny from any to any

# Disable unused services

sudo systemctl disable apache2

sudo systemctl disable telnet

echo "Server hardening completed." | mail -s "Server Hardening Report" admin@example.com

```

**90. Automating Daily System Health Checks:**

* Set up a daily system health check script that checks CPU, memory, disk usage, and sends a report.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Daily system health check

REPORT\_FILE="/tmp/system\_health\_report.txt"

echo "System Health Check - $(date)" > $REPORT\_FILE

echo "-----------------------------" >> $REPORT\_FILE

# CPU Usage

echo "CPU Usage:" >> $REPORT\_FILE

top -bn1 | grep "Cpu(s)" >> $REPORT\_FILE

# Memory Usage

echo "Memory Usage:" >> $REPORT\_FILE

free -m >> $REPORT\_FILE

# Disk Usage

echo "Disk Usage:" >> $REPORT\_FILE

df -h >> $REPORT\_FILE

# Send the health report via email

cat $REPORT\_FILE | mail -s "Daily System Health Check" admin@example.com

```

**91. Automating User Login History Tracking:**

* Track user logins and logouts by monitoring /var/log/auth.log and send alerts if specific users log in or out at unusual times.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Track user logins and send alerts

LOG\_FILE="/var/log/auth.log"

USERS=("user1" "user2")

UNUSUAL\_HOURS=("00:00" "03:00")

for USER in "${USERS[@]}"; do

for HOUR in "${UNUSUAL\_HOURS[@]}"; do

grep "$USER" $LOG\_FILE | grep "$HOUR" > /tmp/unusual\_login\_$USER.log

if [ -s /tmp/unusual\_login\_$USER.log ]; then

echo "Unusual login detected for $USER at $HOUR" | mail -s "Unusual Login Alert for $USER" admin@example.com

fi

done

done

```

**92. Automating AWS EC2 Instance Backup:**

* Automate the creation of snapshots for AWS EC2 instances, providing backups for disaster recovery.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Create snapshot of an EC2 instance

INSTANCE\_ID="i-xxxxxxxxxxxxxxxxx"

SNAPSHOT\_DESCRIPTION="Automated snapshot"

SNAPSHOT\_NAME="snapshot-$(date +'%Y%m%d%H%M%S')"

aws ec2 create-snapshot --instance-id $INSTANCE\_ID --description "$SNAPSHOT\_DESCRIPTION" --tag Name="$SNAPSHOT\_NAME"

echo "Snapshot for instance $INSTANCE\_ID created successfully." | mail -s "EC2 Snapshot Report" admin@example.com

```

**93. Automating Database Table Cleanup:**

* Automatically clean up old records from a database table that are older than a certain number of days.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Clean up old records from a MySQL table

DB\_NAME="my\_database"

DB\_USER="root"

DB\_PASSWORD="password"

TABLE\_NAME="my\_table"

DAYS\_OLD=30

mysql -u $DB\_USER -p$DB\_PASSWORD $DB\_NAME -e "

DELETE FROM $TABLE\_NAME WHERE created\_at < NOW() - INTERVAL $DAYS\_OLD DAY;

"

echo "Old records removed from $TABLE\_NAME older than $DAYS\_OLD days." | mail -s "Database Cleanup Report" admin@example.com

```

**94. Automating Email Alerts for Server Resource Utilization:**

* Set up an email alert system that triggers when server resources (CPU, RAM, disk) exceed a certain threshold.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Check CPU, memory, and disk usage, and send alert if threshold exceeded

CPU\_THRESHOLD=80

MEM\_THRESHOLD=75

DISK\_THRESHOLD=85

CPU\_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.\*, \*\([0-9.]\*\)%\* id.\*/\1/" | awk '{print 100 - $1}')

MEM\_USAGE=$(free | grep Mem | awk '{print $3/$2 \* 100.0}')

DISK\_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

if [ $(echo "$CPU\_USAGE > $CPU\_THRESHOLD" | bc) -eq 1 ]; then

echo "High CPU usage: $CPU\_USAGE%" | mail -s "High CPU Usage Alert" admin@example.com

fi

if [ $(echo "$MEM\_USAGE > $MEM\_THRESHOLD" | bc) -eq 1 ]; then

echo "High Memory usage: $MEM\_USAGE%" | mail -s "High Memory Usage Alert" admin@example.com

fi

if [ $DISK\_USAGE -gt $DISK\_THRESHOLD ]; then

echo "Disk usage is above threshold: $DISK\_USAGE%" | mail -s "Disk Usage Alert" admin@example.com

fi

```

**95. Automating File Synchronization with Rsync Over SSH:**

* Automate file synchronization between a local and remote server using rsync over SSH, useful for backups or data replication.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Sync local files to remote server using rsync over SSH

LOCAL\_DIR="/path/to/local/directory"

REMOTE\_SERVER="user@remote-server:/path/to/remote/directory"

rsync -avz -e ssh $LOCAL\_DIR $REMOTE\_SERVER

echo "Files synchronized to $REMOTE\_SERVER" | mail -s "File Synchronization Report" admin@example.com

```

**96. Automating Service Start/Stop with Remote Servers:**

* Automate starting or stopping a service on remote servers via SSH.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Start or stop a service on remote servers

SERVICE\_NAME="apache2"

ACTION=$1

SERVERS=("server1" "server2" "server3")

for SERVER in "${SERVERS[@]}"; do

ssh user@$SERVER "sudo systemctl $ACTION $SERVICE\_NAME"

if [ $? -eq 0 ]; then

echo "Service $SERVICE\_NAME $ACTION on $SERVER completed."

else

echo "Failed to $ACTION service $SERVICE\_NAME on $SERVER" | mail -s "Service Action Failed" admin@example.com

fi

done

```

**97. Automating Vulnerability Scanning with OpenVAS:**

* Run automated vulnerability scans using OpenVAS and send reports if vulnerabilities are found.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Run OpenVAS scan and send results if vulnerabilities are found

SCAN\_REPORT="/tmp/openvas\_report.txt"

openvas-start # Start OpenVAS if it's not already running

openvas-check-setup --status > $SCAN\_REPORT

if grep -q "Vulnerabilities found" $SCAN\_REPORT; then

echo "Vulnerabilities detected! Please review the report." | mail -s "OpenVAS Vulnerability Report" admin@example.com

else

echo "No vulnerabilities found."

fi

```

**98. Automating Scheduled Web Application Tests:**

* Schedule automated web application tests (e.g., using curl to check if the website is up) and send alerts if the site is down.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Test if the website is up using curl

WEBSITE="http://example.com"

RESPONSE=$(curl -s -o /dev/null -w "%{http\_code}" $WEBSITE)

if [ "$RESPONSE" -ne 200 ]; then

echo "Website $WEBSITE is down. HTTP status code: $RESPONSE" | mail -s "Website Down Alert" admin@example.com

else

echo "Website $WEBSITE is up."

fi

```

**99. Automating Regular AWS EC2 Instance Restart:**

* Automate the restart of an EC2 instance at regular intervals for maintenance or performance optimization.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Restart EC2 instance at regular intervals

INSTANCE\_ID="i-xxxxxxxxxxxxxxxxx"

aws ec2 reboot-instances --instance-ids $INSTANCE\_ID

echo "EC2 instance $INSTANCE\_ID restarted." | mail -s "EC2 Restart Report" admin@example.com

```

**100. Automating Git Repository Cleanup:**

* Clean up old branches in a Git repository to keep it organized.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Delete merged branches in the Git repository

git fetch --prune

git branch --merged | grep -v '\\*' | xargs git branch -d

echo "Merged branches cleaned up in the Git repository." | mail -s "Git Repository Cleanup" admin@example.com

```

These **real-world tasks** offer a wide range of automation possibilities, helping you manage servers, monitor systems, handle backups, and perform various administrative tasks automatically. These scripts make workflows more efficient and reduce manual efforts in routine maintenance, monitoring, and problem-solving.

4o

**You said:**

continue with more real world tasks

**ChatGPT said:**

Here are **even more real-world shell script tasks** that are useful for automating daily administrative operations and managing systems effectively:

**101. Automating File Permissions and Ownership Fixes:**

* Automatically fix file permissions and ownership for a specific directory, ensuring only authorized users have access.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Fix file permissions and ownership

TARGET\_DIR="/var/www/myapp"

USER="www-data"

GROUP="www-data"

# Change ownership

sudo chown -R $USER:$GROUP $TARGET\_DIR

# Set directory permissions to 755 and file permissions to 644

sudo find $TARGET\_DIR -type d -exec chmod 755 {} \;

sudo find $TARGET\_DIR -type f -exec chmod 644 {} \;

echo "Permissions and ownership fixed for $TARGET\_DIR." | mail -s "File Permissions Fix" admin@example.com

```

**102. Automating File Compression and Archiving:**

* Compress files or directories into a .tar.gz archive to reduce storage size or facilitate file transfers.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Compress directory into a tar.gz archive

SOURCE\_DIR="/path/to/source"

BACKUP\_DIR="/path/to/backup"

ARCHIVE\_NAME="backup\_$(date +'%Y%m%d%H%M%S').tar.gz"

tar -czf $BACKUP\_DIR/$ARCHIVE\_NAME -C $SOURCE\_DIR .

echo "Backup created and compressed into $ARCHIVE\_NAME." | mail -s "File Compression Report" admin@example.com

```

**103. Automating Server Reboot After Maintenance:**

* Schedule a server reboot after applying maintenance updates, like security patches or configuration changes.

go

CopyEdit

```bash

pgsql

CopyEdit

#!/bin/bash

# Perform maintenance and reboot server

sudo apt-get update -y

sudo apt-get upgrade -y

sudo apt-get autoremove -y

# Reboot server after maintenance

sudo reboot

echo "Server rebooted after maintenance." | mail -s "Maintenance and Reboot Report" admin@example.com

```

**104. Automating File Integrity Monitoring:**

* Monitor specific directories for unauthorized file changes and alert administrators if any changes are detected.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor file integrity in a directory

MONITORED\_DIR="/var/www/myapp"

CHECKSUM\_FILE="/tmp/checksum\_file.txt"

# Create initial checksum if it doesn't exist

if [ ! -f $CHECKSUM\_FILE ]; then

find $MONITORED\_DIR -type f -exec sha256sum {} \; > $CHECKSUM\_FILE

fi

# Compare current checksums with the stored ones

CURRENT\_CHECKSUM=$(find $MONITORED\_DIR -type f -exec sha256sum {} \;)

STORED\_CHECKSUM=$(cat $CHECKSUM\_FILE)

if [ "$CURRENT\_CHECKSUM" != "$STORED\_CHECKSUM" ]; then

echo "File integrity breach detected in $MONITORED\_DIR!" | mail -s "File Integrity Alert" admin@example.com

else

echo "File integrity intact."

fi

```

**105. Automating Regular Docker Container Cleanup:**

* Clean up unused Docker images, containers, and volumes to free up disk space.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Clean up unused Docker containers, images, and volumes

# Remove stopped containers

docker container prune -f

# Remove unused images

docker image prune -a -f

# Remove unused volumes

docker volume prune -f

echo "Docker cleanup completed." | mail -s "Docker Cleanup Report" admin@example.com

```

**106. Automating Log File Rotation with Compression:**

* Automatically rotate log files to prevent them from becoming too large, and compress old logs.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Rotate and compress log files

LOG\_DIR="/var/log/myapp"

LOG\_FILE="app.log"

BACKUP\_DIR="/backup/logs"

TIMESTAMP=$(date +'%Y%m%d')

mv $LOG\_DIR/$LOG\_FILE $BACKUP\_DIR/$LOG\_FILE.$TIMESTAMP

gzip $BACKUP\_DIR/$LOG\_FILE.$TIMESTAMP

touch $LOG\_DIR/$LOG\_FILE

echo "Log rotation completed and old logs compressed." | mail -s "Log Rotation Report" admin@example.com

```

**107. Automating Server Resource Monitoring with Alerts:**

* Continuously monitor server resource utilization (CPU, memory, disk space) and send alerts if any resource exceeds a certain threshold.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor CPU, memory, and disk usage

CPU\_THRESHOLD=80

MEM\_THRESHOLD=75

DISK\_THRESHOLD=90

# Get CPU, memory, and disk usage

CPU\_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.\*, \*\([0-9.]\*\)%\* id.\*/\1/" | awk '{print 100 - $1}')

MEM\_USAGE=$(free | grep Mem | awk '{print $3/$2 \* 100.0}')

DISK\_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

# Check if usage exceeds threshold

if [ $(echo "$CPU\_USAGE > $CPU\_THRESHOLD" | bc) -eq 1 ]; then

echo "High CPU usage detected: $CPU\_USAGE%" | mail -s "CPU Usage Alert" admin@example.com

fi

if [ $(echo "$MEM\_USAGE > $MEM\_THRESHOLD" | bc) -eq 1 ]; then

echo "High Memory usage detected: $MEM\_USAGE%" | mail -s "Memory Usage Alert" admin@example.com

fi

if [ $DISK\_USAGE -gt $DISK\_THRESHOLD ]; then

echo "High Disk usage detected: $DISK\_USAGE%" | mail -s "Disk Usage Alert" admin@example.com

fi

```

**108. Automating Backup Verification:**

* Verify the integrity of backup files and ensure that the backup has been successfully created.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Verify backup files

BACKUP\_DIR="/backup/mysql"

BACKUP\_FILE="db\_backup\_$(date +'%Y%m%d').sql.gz"

if [ -f "$BACKUP\_DIR/$BACKUP\_FILE" ]; then

echo "Backup file $BACKUP\_FILE exists and is verified." | mail -s "Backup Verification Report" admin@example.com

else

echo "Backup file $BACKUP\_FILE is missing!" | mail -s "Backup Verification Failed" admin@example.com

fi

```

**109. Automating Disk Space Monitoring:**

* Monitor disk usage and send an alert when the disk space exceeds a specified threshold.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor disk space

THRESHOLD=90

DISK\_USAGE=$(df / | grep / | awk '{print $5}' | sed 's/%//g')

if [ $DISK\_USAGE -gt $THRESHOLD ]; then

echo "Warning: Disk space usage is at $DISK\_USAGE%" | mail -s "Disk Space Alert" admin@example.com

else

echo "Disk space usage is below $THRESHOLD%."

fi

```

**110. Automating Scheduled Server Performance Reports:**

* Automatically generate and send a performance report (including CPU, memory, disk usage) to administrators.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Generate server performance report

REPORT\_FILE="/tmp/performance\_report.txt"

echo "Server Performance Report - $(date)" > $REPORT\_FILE

echo "-----------------------------" >> $REPORT\_FILE

echo "CPU Usage:" >> $REPORT\_FILE

top -bn1 | grep "Cpu(s)" >> $REPORT\_FILE

echo "Memory Usage:" >> $REPORT\_FILE

free -m >> $REPORT\_FILE

echo "Disk Usage:" >> $REPORT\_FILE

df -h >> $REPORT\_FILE

cat $REPORT\_FILE | mail -s "Server Performance Report" admin@example.com

```

**111. Automating Automated Web Application Vulnerability Scanning:**

* Set up a script to run a web application vulnerability scanner like OWASP ZAP, Nikto, or others and report if vulnerabilities are found.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Run OWASP ZAP to scan a web application

TARGET\_URL="http://example.com"

OUTPUT\_FILE="/tmp/zap\_report.html"

# Run the ZAP scan

zap-cli quick-scan --url $TARGET\_URL --output $OUTPUT\_FILE

# Check if vulnerabilities are found

if grep -q "vulnerabilities" $OUTPUT\_FILE; then

echo "Vulnerabilities found on $TARGET\_URL!" | mail -s "OWASP ZAP Scan Report" admin@example.com

else

echo "No vulnerabilities found on $TARGET\_URL."

fi

```

**112. Automating User Logout Tracking:**

* Automatically log the time of user logins and logouts, useful for audit or security monitoring.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Track user logins and logouts

LOG\_FILE="/var/log/user\_login\_logout.log"

last -F | awk '{print $1, $4, $5, $6, $7, $8}' > $LOG\_FILE

echo "User login/logout activity tracked." | mail -s "User Activity Report" admin@example.com

```

**113. Automating DNS Record Changes with API:**

* Automate DNS record updates using a service provider's API (such as AWS Route 53, Cloudflare, or others).

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Automate DNS record update

ZONE\_ID="your\_zone\_id"

DNS\_RECORD\_NAME="example.com"

NEW\_IP="1.2.3.4"

# AWS Route 53 API example

aws route53 change-resource-record-sets --hosted-zone-id $ZONE\_ID --change-batch '{

"Changes": [

{

"Action": "UPSERT",

"ResourceRecordSet": {

"Name": "'$DNS\_RECORD\_NAME'",

"Type": "A",

"TTL": 60,

"ResourceRecords": [{"Value": "'$NEW\_IP'"}]

}

}

]

}'

echo "DNS record for $DNS\_RECORD\_NAME updated to $NEW\_IP." | mail -s "DNS Record Update" admin@example.com

```

**114. Automating Server System Reboots After Software Updates:**

* Schedule an automatic reboot of the server after system or software updates have been applied.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Schedule a reboot after updates

sudo apt-get update -y

sudo apt-get upgrade -y

sudo apt-get dist-upgrade -y

sudo reboot

echo "Server scheduled to reboot after updates." | mail -s "Server Update and Reboot" admin@example.com

```

These **real-world tasks** are essential for automating repetitive but crucial tasks like backups, security scans, performance monitoring, and system maintenance. Automation reduces the chances of human error, saves time, and enhances overall system reliability.

4o

**You said:**

continue with more real world tasks that is important and used in shell script

**ChatGPT said:**

Here are **additional real-world shell script tasks** that are highly important and widely used in system administration, cloud management, security, and more:

**115. Automating System Security Patching:**

* Automatically update the system with the latest security patches and restart critical services as needed.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Automate security patching for system updates

# Update the package list

sudo apt-get update -y

# Upgrade only security-related packages

sudo apt-get upgrade -y --only-upgrade

# Clean up unused packages

sudo apt-get autoremove -y

# Restart any services if necessary (e.g., nginx)

sudo systemctl restart nginx

echo "System patched and security updates applied." | mail -s "Security Patching Report" admin@example.com

```

**116. Automating Database Backup and Transfer to Remote Server:**

* Automatically back up a MySQL database and transfer the backup file to a remote server using scp.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Back up MySQL database and transfer to remote server

DB\_NAME="my\_database"

DB\_USER="root"

DB\_PASSWORD="password"

BACKUP\_DIR="/path/to/backups"

BACKUP\_FILE="$BACKUP\_DIR/db\_backup\_$(date +'%Y%m%d').sql"

REMOTE\_SERVER="user@remote-server:/path/to/remote/backup"

# Create database backup

mysqldump -u $DB\_USER -p$DB\_PASSWORD $DB\_NAME > $BACKUP\_FILE

# Transfer backup to remote server

scp $BACKUP\_FILE $REMOTE\_SERVER

echo "Database backup completed and transferred to remote server." | mail -s "Database Backup Report" admin@example.com

```

**117. Automating Automated Email Reports for System Status:**

* Automatically generate and send a system status report including CPU, memory, disk, and network usage.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Send system status report by email

REPORT\_FILE="/tmp/system\_status\_report.txt"

echo "System Status Report - $(date)" > $REPORT\_FILE

echo "-----------------------------" >> $REPORT\_FILE

# CPU Usage

echo "CPU Usage:" >> $REPORT\_FILE

top -bn1 | grep "Cpu(s)" >> $REPORT\_FILE

# Memory Usage

echo "Memory Usage:" >> $REPORT\_FILE

free -m >> $REPORT\_FILE

# Disk Usage

echo "Disk Usage:" >> $REPORT\_FILE

df -h >> $REPORT\_FILE

# Network Usage

echo "Network Usage:" >> $REPORT\_FILE

ifstat -t 1 1 >> $REPORT\_FILE

cat $REPORT\_FILE | mail -s "System Status Report" admin@example.com

```

**118. Automating User Account Lockouts after Multiple Failed Login Attempts:**

* Lock user accounts after a specific number of failed login attempts for security purposes.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Lock accounts after failed login attempts

MAX\_FAILED\_ATTEMPTS=5

LOCK\_DURATION=3600 # 1 hour lock duration

# Check for failed login attempts in the last hour

failed\_attempts=$(grep "Failed password" /var/log/auth.log | wc -l)

if [ $failed\_attempts -ge $MAX\_FAILED\_ATTEMPTS ]; then

# Lock the account for 1 hour

sudo faillock --user $USER --reset

sudo faillock --user $USER --time $LOCK\_DURATION

echo "User account $USER locked after $failed\_attempts failed login attempts." | mail -s "Account Lockout Alert" admin@example.com

fi

```

**119. Automating Website Availability Monitoring:**

* Automate website availability checks by pinging the website and sending an alert if the site is down.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor website availability

WEBSITE="http://example.com"

STATUS\_CODE=$(curl -s -o /dev/null -w "%{http\_code}" $WEBSITE)

if [ "$STATUS\_CODE" -ne 200 ]; then

echo "Website $WEBSITE is down! HTTP status code: $STATUS\_CODE" | mail -s "Website Down Alert" admin@example.com

else

echo "Website $WEBSITE is up and running."

fi

```

**120. Automating Server Log Rotation with Compression:**

* Rotate logs, compress them, and remove old log files to keep the system clean and prevent disk space from filling up.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Rotate and compress logs

LOG\_DIR="/var/log/myapp"

BACKUP\_DIR="/path/to/backup"

LOG\_FILE="app.log"

TIMESTAMP=$(date +'%Y%m%d')

# Rotate the log file

mv $LOG\_DIR/$LOG\_FILE $BACKUP\_DIR/$LOG\_FILE.$TIMESTAMP

touch $LOG\_DIR/$LOG\_FILE

# Compress old logs

gzip $BACKUP\_DIR/$LOG\_FILE.$TIMESTAMP

echo "Log rotation and compression completed." | mail -s "Log Rotation Report" admin@example.com

```

**121. Automating Cloud VM Snapshot with AWS CLI:**

* Automatically create snapshots of virtual machines in AWS EC2 for backup purposes.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Create an EC2 snapshot for backup

INSTANCE\_ID="i-xxxxxxxxxxxxxxxxx"

DESCRIPTION="Automated EC2 Snapshot"

SNAPSHOT\_NAME="snapshot-$(date +'%Y%m%d%H%M%S')"

aws ec2 create-snapshot --instance-id $INSTANCE\_ID --description "$DESCRIPTION" --tag Name="$SNAPSHOT\_NAME"

echo "EC2 snapshot $SNAPSHOT\_NAME created successfully." | mail -s "EC2 Snapshot Report" admin@example.com

```

**122. Automating System Cleanup for Unused Files:**

* Clean up old, unused files from directories to free up disk space and maintain system efficiency.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Clean up old unused files

TARGET\_DIR="/path/to/temp/files"

DAYS=30

find $TARGET\_DIR -type f -mtime +$DAYS -exec rm -f {} \;

echo "Old files older than $DAYS days removed from $TARGET\_DIR." | mail -s "System Cleanup Report" admin@example.com

```

**123. Automating Backup and Restore for Redis:**

* Automatically back up Redis data and restore it when necessary.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Backup Redis database

BACKUP\_DIR="/path/to/backup"

BACKUP\_FILE="$BACKUP\_DIR/redis\_backup\_$(date +'%Y%m%d').rdb"

# Dump Redis data to a backup file

redis-cli SAVE

cp /var/lib/redis/dump.rdb $BACKUP\_FILE

echo "Redis database backed up to $BACKUP\_FILE." | mail -s "Redis Backup Report" admin@example.com

# Restore Redis database (example)

# redis-cli shutdown

# cp /path/to/backup/redis\_backup.rdb /var/lib/redis/dump.rdb

# redis-server

```

**124. Automating Daily Email Notification with System Uptime:**

* Send a daily email with the system's uptime and basic statistics.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Send daily email with system uptime and stats

UPTIME=$(uptime -p)

LOAD\_AVG=$(uptime | awk -F'load average: ' '{print $2}')

MEMORY\_USAGE=$(free -m | grep Mem | awk '{print $3/$2 \* 100.0}')

DISK\_USAGE=$(df / | grep / | awk '{print $5}' | sed 's/%//g')

echo "System Uptime: $UPTIME" > /tmp/daily\_report.txt

echo "Load Average: $LOAD\_AVG" >> /tmp/daily\_report.txt

echo "Memory Usage: $MEMORY\_USAGE%" >> /tmp/daily\_report.txt

echo "Disk Usage: $DISK\_USAGE%" >> /tmp/daily\_report.txt

cat /tmp/daily\_report.txt | mail -s "Daily System Stats" admin@example.com

```

**125. Automating Scheduled Application Deployment with Git:**

* Automate the deployment of a new version of an application from Git repository by pulling the latest code and restarting the web server.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Deploy new application version from Git and restart web server

APP\_DIR="/var/www/myapp"

GIT\_REPO="https://github.com/username/myapp.git"

SERVICE\_NAME="nginx"

# Pull latest changes from Git repository

cd $APP\_DIR

git pull $GIT\_REPO

# Restart the web server

sudo systemctl restart $SERVICE\_NAME

echo "Deployment completed, $SERVICE\_NAME restarted." | mail -s "Deployment Report" admin@example.com

```

**126. Automating System Time Synchronization:**

* Ensure that all system clocks are synchronized using NTP.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Sync system time using NTP

sudo apt-get install -y ntp

sudo systemctl enable ntp

sudo systemctl start ntp

echo "System time synchronized with NTP." | mail -s "NTP Synchronization Report" admin@example.com

```

**127. Automating Failed SSH Login Alert:**

* Monitor failed SSH login attempts and send an alert if an excessive number of failed logins occur within a short period.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor failed SSH logins

MAX\_FAILED\_LOGINS=5

FAILED\_LOGINS=$(grep "Failed password" /var/log/auth.log | wc -l)

if [ $FAILED\_LOGINS -ge $MAX\_FAILED\_LOGINS ]; then

echo "Excessive failed SSH logins detected: $FAILED\_LOGINS" | mail -s "Failed SSH Login Alert" admin@example.com

fi

```

These **real-world shell script tasks** are designed to automate essential operations, security tasks, backups, and system monitoring in a way that ensures system reliability and reduces manual intervention. They are critical for system administrators and DevOps engineers who manage servers, databases, web applications, and cloud infrastructure.

4o

**You said:**

continue with more unique tasks

**ChatGPT said:**

Here are **more unique real-world shell script tasks** that are useful for various automation scenarios, system maintenance, and cloud operations:

**128. Automating File Synchronization Across Multiple Locations:**

* Automatically synchronize files between multiple servers, useful for mirroring data or maintaining consistency across different environments.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Synchronize files across multiple servers

SOURCE\_DIR="/path/to/source"

DEST\_DIRS=("user@server1:/path/to/destination" "user@server2:/path/to/destination")

for DEST in "${DEST\_DIRS[@]}"; do

rsync -avz --delete $SOURCE\_DIR $DEST

if [ $? -eq 0 ]; then

echo "Files synchronized to $DEST successfully."

else

echo "Failed to synchronize files to $DEST" | mail -s "File Synchronization Error" admin@example.com

fi

done

```

**129. Automating SSL/TLS Certificate Expiration Monitoring:**

* Automatically check the expiration date of SSL/TLS certificates for your domains and send an alert when the certificate is nearing expiry.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Check SSL/TLS certificate expiration date

DOMAIN="example.com"

CERT\_EXPIRY\_DATE=$(echo | openssl s\_client -connect $DOMAIN:443 2>/dev/null | openssl x509 -noout -enddate)

EXPIRY\_DATE=$(echo $CERT\_EXPIRY\_DATE | sed 's/notAfter=//')

EXPIRY\_TIMESTAMP=$(date -d "$EXPIRY\_DATE" +%s)

CURRENT\_TIMESTAMP=$(date +%s)

DIFF=$((($EXPIRY\_TIMESTAMP - $CURRENT\_TIMESTAMP)/86400))

if [ $DIFF -le 30 ]; then

echo "SSL certificate for $DOMAIN is expiring in $DIFF days!" | mail -s "SSL Certificate Expiry Alert" admin@example.com

fi

```

**130. Automating Redis Data Backup with Cron:**

* Automate the backup of Redis data using the BGSAVE command and schedule it via cron for regular backups.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Backup Redis data

BACKUP\_DIR="/path/to/redis/backup"

BACKUP\_FILE="redis\_backup\_$(date +'%Y%m%d%H%M%S').rdb"

# Trigger Redis backup

redis-cli BGSAVE

# Copy the RDB file to backup directory

cp /var/lib/redis/dump.rdb $BACKUP\_DIR/$BACKUP\_FILE

echo "Redis backup created: $BACKUP\_FILE" | mail -s "Redis Backup Report" admin@example.com

```

**131. Automating Monitoring of Server Reboots:**

* Track when a server was last rebooted and send an email notification if the server has been up for too long without a reboot.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor server uptime and send alert if no reboot in a while

UPTIME=$(uptime -p)

LAST\_REBOOT=$(who -b | awk '{print $3, $4}')

# Check if the system has been up for more than 30 days

if [[ $(uptime -s | awk '{print $1}') > $(date --date='30 days ago' +%Y-%m-%d) ]]; then

echo "The server has been up since $LAST\_REBOOT without a reboot for over 30 days." | mail -s "Server Reboot Alert" admin@example.com

fi

```

**132. Automating Local Database to Cloud Database Synchronization:**

* Automatically synchronize a local MySQL database with a remote MySQL database hosted in the cloud (e.g., AWS RDS).

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Synchronize local MySQL database with cloud database

LOCAL\_DB="local\_db"

REMOTE\_DB="cloud\_db"

DB\_USER="root"

DB\_PASSWORD="password"

REMOTE\_HOST="cloud-db-hostname.amazonaws.com"

# Dump the local database

mysqldump -u $DB\_USER -p$DB\_PASSWORD $LOCAL\_DB > /tmp/local\_db\_backup.sql

# Sync the backup to the cloud database

mysql -u $DB\_USER -p$DB\_PASSWORD -h $REMOTE\_HOST $REMOTE\_DB < /tmp/local\_db\_backup.sql

echo "Database synchronization completed from local to cloud." | mail -s "Database Sync Report" admin@example.com

```

**133. Automating Image Compression for Web Applications:**

* Automatically compress images in a directory to reduce the load time for web applications.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Compress images in a directory

IMAGE\_DIR="/path/to/images"

COMPRESSED\_DIR="/path/to/compressed\_images"

for IMAGE in $IMAGE\_DIR/\*.{jpg,png,gif}; do

convert $IMAGE -quality 85 $COMPRESSED\_DIR/$(basename $IMAGE)

done

echo "Image compression completed for directory: $IMAGE\_DIR" | mail -s "Image Compression Report" admin@example.com

```

**134. Automating a Website’s Content Backup (via wget):**

* Use wget to create an offline backup of a website, storing all HTML and media files.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Backup website content

WEBSITE\_URL="http://example.com"

BACKUP\_DIR="/path/to/backup"

# Download website content using wget

wget --mirror --convert-links --adjust-extension --wait=1 --limit-rate=200k $WEBSITE\_URL -P $BACKUP\_DIR

echo "Website content backup completed for $WEBSITE\_URL." | mail -s "Website Backup Report" admin@example.com

```

**135. Automating File Integrity Check with aide:**

* Use aide (Advanced Intrusion Detection Environment) to monitor file integrity and detect any unauthorized changes.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Run AIDE for file integrity check

AIDE\_DB="/var/lib/aide/aide.db.new"

AIDE\_CONF="/etc/aide/aide.conf"

# Initialize the AIDE database

sudo aideinit --conf=$AIDE\_CONF --dbpath=$AIDE\_DB

# Compare the current state with the AIDE database

sudo aide --check --dbpath=$AIDE\_DB

# Send alert if changes are detected

if [ $? -ne 0 ]; then

echo "AIDE integrity check failed, unauthorized changes detected." | mail -s "File Integrity Alert" admin@example.com

fi

```

**136. Automating MySQL Replication Health Check:**

* Check if MySQL replication is running smoothly, and alert if there are replication errors or lags.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Check MySQL replication status

REPLICATION\_STATUS=$(mysql -u root -p"password" -e "SHOW SLAVE STATUS\G" | grep -i 'Slave\_IO\_Running\|Slave\_SQL\_Running')

if [[ "$REPLICATION\_STATUS" != \*"Yes"\* ]]; then

echo "MySQL replication is down!" | mail -s "MySQL Replication Alert" admin@example.com

else

echo "MySQL replication is running smoothly."

fi

```

**137. Automating Disk Partition Backup:**

* Create a backup of a disk partition using dd, useful for disaster recovery scenarios.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Create disk partition backup using dd

SOURCE\_PARTITION="/dev/sda1"

BACKUP\_FILE="/path/to/backup/disk\_backup\_$(date +'%Y%m%d').img"

# Create a backup of the partition

sudo dd if=$SOURCE\_PARTITION of=$BACKUP\_FILE bs=64K conv=noerror,sync

echo "Disk partition $SOURCE\_PARTITION backed up to $BACKUP\_FILE." | mail -s "Disk Backup Report" admin@example.com

```

**138. Automating AWS EC2 Instance Scaling Based on CPU Utilization:**

* Automatically scale AWS EC2 instances based on CPU utilization thresholds using AWS CLI.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Automatically scale EC2 instances based on CPU usage

INSTANCE\_ID="i-xxxxxxxxxxxxxxxxx"

CPU\_USAGE\_THRESHOLD=80

CPU\_USAGE=$(aws cloudwatch get-metric-statistics --metric-name CPUUtilization --start-time $(date -u -d "5 minutes ago" +'%Y-%m-%dT%H:%M:%SZ') --end-time $(date -u +'%Y-%m-%dT%H:%M:%SZ') --period 300 --statistics Average --dimensions Name=InstanceId,Value=$INSTANCE\_ID --query 'Datapoints[0].Average')

if (( $(echo "$CPU\_USAGE > $CPU\_USAGE\_THRESHOLD" | bc -l) )); then

# Scale up the instance (e.g., increase instance size)

aws ec2 modify-instance-type --instance-id $INSTANCE\_ID --instance-type t2.large

echo "Scaling up instance $INSTANCE\_ID due to high CPU usage ($CPU\_USAGE%)" | mail -s "EC2 Instance Scaling Alert" admin@example.com

fi

```

**139. Automating Network Bandwidth Usage Monitoring:**

* Monitor network bandwidth usage and send alerts if the bandwidth usage exceeds a set threshold.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor network bandwidth usage

INTERFACE="eth0"

BANDWIDTH\_THRESHOLD=1000 # 1000 Kbps

# Get current network usage (in Kbps)

BANDWIDTH\_USAGE=$(ifstat -i $INTERFACE 1 1 | awk 'NR==3 {print $1}')

if (( $(echo "$BANDWIDTH\_USAGE > $BANDWIDTH\_THRESHOLD" | bc -l) )); then

echo "High bandwidth usage detected: $BANDWIDTH\_USAGE Kbps" | mail -s "Bandwidth Alert" admin@example.com

fi

```

**140. Automating Cloud Infrastructure Cost Monitoring (AWS):**

* Monitor your AWS infrastructure costs and send an alert when your bill exceeds a set threshold.

go

CopyEdit

```bash

bash

CopyEdit

#!/bin/bash

# Monitor AWS costs using AWS Cost Explorer

COST\_THRESHOLD=500 # Set threshold in USD

CURRENT\_COST=$(aws ce get-cost-and-usage --time-period Start=$(date +'%Y-%m-01'),End=$(date +'%Y-%m-31') --granularity MONTHLY --metrics "BlendedCost" --query 'ResultsByTime[0].Total.BlendedCost.Amount' --output text)

if (( $(echo "$CURRENT\_COST > $COST\_THRESHOLD" | bc -l) )); then

echo "AWS cost exceeded threshold: $CURRENT\_COST USD" | mail -s "AWS Cost Alert" admin@example.com

fi

```

These **unique real-world shell script tasks** offer solutions for managing systems, scaling cloud infrastructure, ensuring security, and maintaining uptime across multiple environments. These scripts are practical and essential for IT administrators, cloud engineers, security teams, and DevOps professionals to automate repetitive tasks, monitor critical resources, and ensure smooth operations.